

ISIS-II MCS-86 ASSEMBLER V1.0 ASSEMBLY OF MODULE DISL86
 OBJECT MODULE PLACED IN DISL86.OBJ
 ASSEMBLER INVOKED BY: ASM86 DISL86.ASM DEBUG

LOC	OBJ	LINE	SOURCE
		1	; transient interface module for 8086
		2	; JULY, 1980
		3	dgroup group dats
		4	dats segment public 'DATS'
0000	3F	5	db '?'
		6	dats ends
		7	cgroup group code
		8	extrn disem:near
		9	public boot
		10	public conin,conout
		11	code segment public 'CODE'
		12	assume cs:cgroup
0000		13	disent proc
1400		14	org 1400h
1400	E90000	15	jmp disem
		16	disent endp
1403		17	boot proc
1403	C3	18	ret
		19	boot endp
1404		20	conin proc
1404	E9FCEC	21	jmp \$-1301h
		22	conin endp
1407		23	conout proc
1407	E9FCEC	24	jmp \$-1301h
		25	conout endp
		26	code ends
		27	end

CP/M-86 V.1.1 (Vol. III)
 COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # C81-162

SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES
??SEG	. SEGMENT		SIZE=0000H PARA PUBLIC
BOOT.	. L NEAR	1403H	CODE PUBLIC
CGROUP.	GROUP		CODE
CODE.	. SEGMENT		SIZE=140AH PARA PUBLIC 'CODE'
CONIN.	. L NEAR	1404H	CODE PUBLIC
CONOUT.	. L NEAR	1407H	CODE PUBLIC
DATS.	. SEGMENT		SIZE=0001H PARA PUBLIC 'DATS'
DGROUP.	GROUP		DATS
DISEH.	. L NEAR	0000H	EXTRN
DISENT.	. L NEAR	0000H	CODE

ASSEMBLY COMPLETE, NO ERRORS FOUND

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE DISEM86
 OBJECT MODULE PLACED IN DIS86.OBJ
 COMPILER INVOKED BY: PLM86 DIS86.PLH DEBUG PAGESWIDTH(100) XREF

```

$title('8086 disassembler')
$date(5/14/81)
$compact
$optimize(2)

1      disem86: do;

2      1      declare
              cr literally '0dh',
              lf literally '0ah',
              true literally '1',
              false literally '0';

              $include(optab.dat)

3      1      = declare ops2 ($) byte initial (
              =      'IN', 'JA', 'JB', 'JC', 'JE',
              =      'JG', 'JL', 'JO', 'JP', 'JS',
              =      'JZ', 'OR');
              =

4      1      = declare ops3 ($) byte initial (
              =      'AAA', 'AAD', 'AAM', 'AAS', 'ADC',
              =      'ADD', 'AND', 'CBW', 'CLC', 'CLD',
              =      'CLI', 'CMC', 'CMP', 'CS:', 'CWD',
              =      'DAA', 'DAS', 'DEC', 'DIV', 'DS:',
              =      'ES:', 'ESC', 'HLT', 'INC', 'INT',
              =      'JAE', 'JBE', 'JGE', 'JLE', 'JMP',
              =      'JNA', 'JNB', 'JNC', 'JNE', 'JNG',
              =      'JNL', 'JNO', 'JNP', 'JNS', 'JNZ',
              =      'JPE', 'JPO', 'LDS', 'LEA', 'LES',
              =      'MOV', 'MUL', 'NEG', 'NOP', 'NOT',
              =      'OUT', 'POP', 'RCL', 'RCR', 'REP',
              =      'RET', 'ROL', 'ROR', 'SAL', 'SAR',
              =      'SBB', 'SHL', 'SHR', 'SS:', 'STC',
              =      'STD', 'STI', 'SUB', 'XOR');
              =

5      1      = declare ops4 ($) byte initial (
              =      'CALL', 'IDIV', 'IMUL', 'INTO', 'IRET',
              =      'JCXZ', 'JMPF', 'JNPS', 'JNAE', 'JNBE',
              =      'JNGE', 'JNLE', 'LAHF', 'LOCK', 'LOOP',
              =      'POPF', 'PUSH', 'REPE', 'REPZ', 'RETF',
              =      'SAHF', 'TEST', 'WAIT', 'XCHG', 'XLAT');
              =

6      1      = declare ops5 ($) byte initial (
              =      'CALLF', 'CMPSB', 'CHPSW', 'LODSB', 'LODSW',
              =      'LOOPE', 'LOOPZ', 'MOVSF', 'MOVSW', 'PUSHF',
              =      'REPNE', 'REPZ', 'SCASB', 'SCASW', 'STOSB',
              =      'STOSW');
              =

7      1      = declare ops6 ($) byte initial (
              =      'LOOPNE', 'LOOPNZ');
  
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

      =
8  1  = declare nops (5) byte public initial (12, 69, 25, 16, 2);
      =
9  1  = declare opn$in (*) byte public initial (0, 12, 81, 106, 122, 255);
      = $

10 1  declare
      tab$ptrs (5) address public initial (.ops2, .ops3, .ops4, .ops5, .ops6);

11 1  declare
      left$bracket byte data ('['),
      right$bracket byte data (']');

12 1  declare
      alt$table$base address,
      alt$table based alt$table$base (16) byte,
      alt$table$ptrs (8) address external;

13 1  declare
      mod$bits byte,
      reg$bits byte,
      rm$bits byte,
      byte1$reg$bits byte,
      d$bit byte,
      s$bit byte,
      v$bit byte,
      w$bit byte,
      z$bit byte;

14 1  declare
      mnemonic$index byte, /* index into opcodes */
      instr$type byte,
      table$ptr address,
      table$char based table$ptr byte,
      disem$ptr pointer,
      disem$offset address at (.disem$ptr),
      disem$end address,
      disem$byte based disem$ptr (1) byte,
      disem$word based disem$ptr (1) address,
      b$or$w$flag byte,
      error$flag byte;

15 1  declare instr$table (512) byte external;

16 1  declare
      ax$reg literally '0',
      cx$reg literally '1',
      dx$reg literally '2',
      bx$reg literally '3',
      sp$reg literally '4',
      bp$reg literally '5',
      si$reg literally '6',
      di$reg literally '7';

17 1  declare
      al$reg literally '0',
      cl$reg literally '1',

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

        dl$reg literally '2',
        bl$reg literally '3',
        ah$reg literally '4',
        ch$reg literally '5',
        dh$reg literally '6',
        bh$reg literally '7';

18 1    declare
        es$reg literally '0',
        cs$reg literally '1',
        ss$reg literally '2',
        ds$reg literally '3';

19 1    declare
        reg16 ($) byte public initial ('AX', 'CX', 'DX', 'E', 'SP', 'BP', 'SI', 'DI'),
        reg8 ($) byte public initial ('AL', 'CL', 'DL', 'BL', 'AH', 'CH', 'DH', 'BH'),
        segreg ($) byte public initial ('ES', 'CS', 'SS', 'DS');

20 1    conout: procedure (c) external;
21 2    declare c byte;
22 2    end conout;

23 1    comma: procedure;
24 2    call conout (',');
25 2    end comma;

26 1    printa: procedure (a) PUBLIC;
27 2    declare a address;
28 2    declare b based a byte;
29 2    do while b <> '$';
30 3        call conout (b);
31 3        a = a + 1;
32 3    end;
33 2    end printa;

34 1    print$nibble: procedure (b);
35 2    declare b byte;
36 2    if b > 9 then call conout (b - 10 + 'A');
38 2    else call conout (b + '0');
39 2    end print$nibble;

40 1    print$byte: procedure (b);
41 2    declare b byte;
42 2    call print$nibble (shr (b, 4));
43 2    call print$nibble (b and 0fh);
44 2    end print$byte;

45 1    print$word: procedure (a) public;
46 2    declare a address;
47 2    call print$byte (high (a));
48 2    call print$byte (low (a));
49 2    end print$word;

50 1    error: procedure;
51 2    call printa (('.'??= $'));
52 2    call print$byte (disem$byte (0));
53 2    disem$offset = disem$offset + 1;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
34 2      end error;

55 1      set$bits: procedure;
56 2          byte1$reg$bits = disem$byte (0) and 7;
57 2          mod$bits = shr (disem$byte (1), 6);
58 2          reg$bits = shr (disem$byte (1), 3) and 7;
59 2          r$bits = disem$byte (1) and 7;
60 2          w$bit, z$bit = disem$byte (0) and 1;
61 2          d$bit, s$bit, v$bit = shr (disem$byte (0), 1) and 1;
62 2          end set$bits;

63 1      print$b$or$w: procedure;
64 2          if w$bit then call printa (., ('WORD $'));
66 2          else call printa (., ('BYTE $'));
67 2          end print$b$or$w;

68 1      print$reg: procedure (reg$add, reg);
69 2          declare reg$add address, reg byte;
70 2          table$ptr = reg$add + shl (reg, 1);
71 2          call conout (table$char);
72 2          table$ptr = table$ptr + 1;
73 2          call conout (table$char);
74 2          end print$reg;

75 1      print$reg8: procedure (reg);
76 2          declare reg byte;
77 2          call print$reg (., reg8, reg);
78 2          end print$reg8;

79 1      print$reg16: procedure (reg);
80 2          declare reg byte;
81 2          call print$reg (., reg16, reg);
82 2          end print$reg16;

83 1      print$reg8$or$16: procedure (reg$num);
84 2          declare reg$num byte;
85 2          if w$bit then call print$reg16 (reg$num);
87 2          else call print$reg8 (reg$num);
88 2          end print$reg8$or$16;

89 1      print$2$reg16: procedure (r1, r2);
90 2          declare (r1, r2) byte;
91 2          call print$reg16 (r1);
92 2          call conout ('+');
93 2          call print$reg16 (r2);
94 2          end print$2$reg16;

95 1      print$a$reg: procedure;
96 2          if w$bit then call print$reg16 (ax$reg);
98 2          else call print$reg8 (ax$reg);
99 2          end print$a$reg;

100 1      print$seg$reg: procedure (reg);
101 2          declare reg byte;
102 2          call print$reg (., seg$reg, reg);
103 2          end print$seg$reg;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

```

104 1    print$data$8: procedure;
105 2        call print$byte (disem$byte (0));
106 2        disem$offset = disem$offset + 1;
107 2        end print$data$8;

108 1    print$data$16: procedure;
109 2        call print$word (disem$word (0));
110 2        disem$offset = disem$offset + 2;
111 2        end print$data$16;

112 1    print$data$8$or$16: procedure;
113 2        if w$bit then call print$data$16;
115 2        else call print$data$8;
116 2        end print$data$8$or$16;

117 1    print$data$sw: procedure;
118 2        if rol (disem$byte (0), 1) then call print$word (0ff00h or disem$byte (0));
120 2        else call print$word (disem$byte (0));
121 2        disem$offset = disem$offset + 1;
122 2        end print$data$sw;

123 1    print$signed$8: procedure;
124 2        declare a address;
125 2        a = disem$byte (0);
126 2        if low (a) >= 80h then a = a or 0ff00h; /* sign extend to 16 bits */
128 2        call print$word (disem$offset + a + 1);
129 2        disem$offset = disem$offset + 1;
130 2        end print$signed$8;

131 1    print$signed$16: procedure;
132 2        call print$word (disem$offset + disem$word (0) + 2);
133 2        disem$offset = disem$offset + 2;
134 2        end print$signed$16;

135 1    print$direct$addr: procedure;
136 2        call conout (left$bracket);
137 2        call print$word (disem$word (0));
138 2        call conout (right$bracket);
139 2        disem$offset = disem$offset + 2;
140 2        end print$direct$addr;

141 1    print$mod$rm: procedure;
142 2        disem$offset = disem$offset + 1; /* point past mod/reg/rm byte */
143 2        if mod$bits = 3 then
144 2            do;
145 3                call print$reg$8$or$16 (rm$bits);
146 3                return;
147 3            end;
148 2        if b$or$w$flag then call print$b$or$w;
150 2        if rm$bits = 6 and mod$bits = 0 then
151 2            do;
152 3                call print$direct$addr;
153 3                return;
154 3            end;
155 2        if mod$bits = 1 then
156 2            do;
157 3                if (rm$bits <> 6) or (disem$byte (0) <> 0)

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

        then call print$byte (disem$byte (0));
157 3      disem$offset = disem$offset + 1;
160 3      end;
161 2      else if mod$bits = 2 then
162 2          do;
163 3              call print$word (disem$word (0));
164 3              disem$offset = disem$offset + 2;
165 3          end;
        call conout (left$bracket);
167 2      do case rm$bits;
168 3          call print$2$reg$16 (3, 6);
169 3          call print$2$reg$16 (3, 7);
170 3          call print$2$reg$16 (5, 6);
171 3          call print$2$reg$16 (5, 7);
172 3          call print$reg$16 (6);
173 3          call print$reg$16 (7);
174 3          call print$reg$16 (5);
175 3          call print$reg$16 (3);
176 3      end;
177 2      call conout (right$bracket);
178 2      end print$mod$rm;

179 1      print$mod$reg$rm: procedure;
180 2          if d$bit then
181 2              do;
182 3                  call print$reg$8$or$16 (reg$bits);
183 3                  call conout (',');
184 3                  call print$mod$rm;
185 3              end;
            else
186 2              do;
187 3                  call print$mod$rm;
188 3                  call conout (',');
189 3                  call print$reg$8$or$16 (reg$bits);
190 3              end;
191 2          end print$mod$reg$rm;

192 1      print$mnemonic: procedure;
193 2          declare (len, i) byte;
194 2          len = 2;
195 2          do while mnemonic$index >= opn$in (len - 1);
196 3              len = len + 1;
197 2          end;
198 2          table$ptr = tab$ptrs (len - 2) + (mnemonic$index - opn$in (len - 2))
            * len;
199 2          do i = 1 to 7;
200 3              if i <= len then
201 3                  do;
202 4                      call conout (table$char);
203 4                      table$ptr = table$ptr + 1;
204 4                  end;
205 3              else call conout (' ');
206 2          end;
207 2          disem$offset = disem$offset + 1;
208 2          end print$mnemonic;

209 1      type1: procedure;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```
210 2      call print$mnemonic;
211 2      end type1;

212 1      type2: procedure;
213 2          if disem$byte (1) = 0ah then
214 2              do;
215 3                  call print$mnemonic;
216 3                  disem$offset = disem$offset + 1;
217 3              end;
218 2          else error$flag = true;
219 2          end type2;
```

```
220 1      type3: procedure;
221 2          call print$mnemonic;
222 2          call print$reg$16 (byte1$reg$bits);
223 2          end type3;
```

```
224 1      type4: procedure;
225 2          declare temp byte;
226 2          temp = shr (disem$byte (0), 3) and 3;
227 2          call print$mnemonic;
228 2          call print$segreg (temp);
229 2          end type4;
```

```
230 1      type5: procedure;
231 2          call print$mnemonic;
232 2          call print$signed$8;
233 2          end type5;
```

```
234 1      type6: procedure;
235 2          call print$mnemonic;
236 2          call print$signed$16;
237 2          end type6;
```

```
238 1      type8: procedure; /* 7, 9 */
239 2          call print$mnemonic;
240 2          call print$mod$rm;
241 2          end type8;
```

```
242 1      type10: procedure;
243 2          call print$mnemonic;
244 2          call print$data$8;
245 2          end type10;
```

```
246 1      type11: procedure;
247 2          call print$mnemonic;
248 2          call print$data$16;
249 2          end type11;
```

```
250 1      type12: procedure;
251 2          call print$mnemonic;
252 2          call conout ('3');
253 2          end type12;
```

```
254 1      type13: procedure;
255 2          declare temp address;
256 2          call print$mnemonic;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
257 2      temp = disem$word (0);
258 2      disem$offset = disem$offset + 2;
259 2      call print$data$16;
260 2      call conout ('');
261 2      call print$word (temp);
262 2      end type13;

263 1      type14: procedure; /* 15, 16, 17 */
264 2          call int$mnemonic;
265 2          call print$mod$reg$rm;
266 2          end type14;

267 1      type18: procedure; /* 19, 20, 21 */
268 2          call print$mnemonic;
269 2          if d$bit then
270 2              do;
271 3              call print$direct$addr;
272 3              call comma;
273 3              call print$A$reg;
274 3          end;
275 2          else
276 2              do;
277 3              call print$A$reg;
278 3              call comma;
279 3              call print$direct$addr;
280 3          end;
281 2          end type18;

281 1      type22: procedure;
282 2          call print$mnemonic;
283 2          if d$bit then
284 2              do;
285 3              call print$data$8;
286 3              call comma;
287 3              call print$A$reg;
288 3          end;
289 2          else
290 2              do;
291 3              call print$A$reg;
292 3              call comma;
293 3              call print$data$8;
294 3          end;
295 2          end type22;

295 1      type23: procedure; /* 24 */
296 2          call print$mnemonic;
297 2          call print$A$reg;
298 2          call comma;
299 2          call print$data$8$or$16;
300 2          end type23;

301 1      type25: procedure; /* 26 */
302 2          call print$mnemonic;
303 2          if d$bit then
304 2              do;
305 3              call print$reg$16 (dx$reg);
306 3              call comma;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

307 3      call print$A$reg;
308 3      end;
      else
309 2      do;
310 3          call print$A$reg;
311 3          call comma;
312 3          call print$reg$16 (dx$reg);
313 3      end;
314 2      end type25;

315 1      type27: procedure; /* 28, 29, 30 */
316 2          call print$anemonic;
317 2          btor$w$flag = true;
318 2          call print$mod$rm;
319 2          call comma;
320 2          if v$bit then call print$reg$8 (cl$reg);
322 2          else call conout ('1');
323 2          end type27;

324 1      type31: procedure; /* 32 */
325 2          call setbits;
326 2          reg$bits = byte1$reg$bits;
327 2          w$bit = shr (disem$byte (0), 3) and 1;
328 2          call print$anemonic;
329 2          call print$reg$8$or$16 (reg$bits);
330 2          call comma;
331 2          call print$data$8$or$16;
332 2          end type31;

333 1      type33: procedure;
334 2          call print$anemonic;
335 2          call print$reg$16 (ax$reg);
336 2          call comma;
337 2          call print$reg$16 (byte1$reg$bits);
338 2          end type33;

339 1      type34: procedure; /* 35 */
340 2          call print$anemonic;
341 2          btor$w$flag = true;
342 2          call print$mod$rm;
343 2          call comma;
344 2          call print$data$8$or$16;
345 2          end type34;

346 1      type36: procedure; /* 37 */
347 2          w$bit = true; /* force 16 bit reg, new */
348 2          if reg$bits > 3 then
349 2              do;
350 3                  error$flag = true;
351 3                  return;
352 3              end;
353 2          call print$anemonic;
354 2          if d$bit then
355 2              do;
356 3                  call print$seg$reg (reg$bits);
357 3                  call comma;
358 3                  call print$mod$rm;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
359 3      end;
      else
360 2      do;
361 3          call print$mod$rm;
362 3          call comma;
363 3          call print$seg$reg (reg$bits);
364 3      end;
365 2      end type36;

366 1      type38: procedure;
367 2          call print$memonic;
368 2          call print$mod$rm;
369 2          call comma;
370 2          call print$data$8;
371 2          end type38;

372 1      type39: procedure;
373 2          if mod$bits = 3 then
374 2              do;
375 3                  error$flag = true;
376 3                  return;
377 3              end;
378 2          call print$memonic;
379 2          call print$reg$16 (reg$bits);
380 2          call comma;
381 2          call print$mod$rm;
382 2          end type39;

383 1      type40: procedure;  /* 41 */
384 2          if mod$bits = 3 then
385 2              do;
386 3                  error$flag = true;
387 3                  return;
388 3              end;
389 2          call print$memonic;
390 2          b$or$w$flag = true;
391 2          call print$mod$rm;
392 2          call comma;
393 2          call print$data$8$or$16;
394 2          end type40;

395 1      type42: procedure;
396 2          call print$memonic;
397 2          call print$byte (shl (byte1$reg$bits, 3) or reg$bits);
398 2          call comma;
399 2          call print$mod$rm;
400 2          end type42;

401 1      type44: procedure;
402 2          call print$memonic;
403 2          b$or$w$flag = true;
404 2          call print$mod$rm;
405 2          call comma;
406 2          if st$bit = 1 and w$bit = 1 then call print$data$sw;
408 2          else call print$data$8$or$16;
409 2          end type44;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
410 1    type45: procedure;
411 2        btor$w$flag = true;
412 2        call type8;
413 2        end type45;

414 1    dis: procedure;
415 2        error$flag, btor$w$flag = false;
416 2        call set$bits;
417 2        if instr$type = 26 then
418 2            do;
419 3                alt$table$base = alt$table$ptrs (mnemonic$index);
420 3                mnemonic$index = alt$table (reg$bits * 2);
421 3                instr$type = alt$table (reg$bits * 2 + 1);
422 3            end;
423 2        if instr$type > 28 then error$flag = true;
        else
425 2            do case instr$type;
426 3                error$flag = true;
427 3                call type1;
428 3                call type2;
429 3                call type3;
430 3                call type4;
431 3                call type5;
432 3                call type6;
433 3                call type8;
434 3                call type10;
435 3                call type11;
436 3                call type12;
437 3                call type13;
438 3                call type14;
439 3                call type18;
440 3                call type22;
441 3                call type23;
442 3                call type25;
443 3                call type27;
444 3                call type31;
445 3                call type33;
446 3                call type34;
447 3                call type36;
448 3                call type38;
449 3                call type39;
450 3                call type40;
451 3                call type42;
452 3                ;
453 3                call type44;
454 3                call type45;
455 3            end;
456 2        if error$flag then call error;
458 2        end dis;
```

```
459 1    disem: procedure (disloc) address public;
460 2        declare disloc pointer;
461 2        declare nprefix byte;
462 2        disem$ptr = disloc;
463 2        nprefix = 0;
464 2        do while true;
465 3            mnemonic$index = instr$table (disem$byte (0) * 2);
```

CP/M-86 v.1.1 (Vol. III)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```
466 3      instr$type = instr$table (disem$byte (0) * 2 + 1);
467 3      if instr$type = 0ffh and nprefix < 3 then
468 3          do;
469 4          call print$memonic;
470 4          nprefix = nprefix + 1;
471 4      end;
      else
472 3          do;
473 4          if instr$type = 0ffh then instr$type = 1;
475 4          call dis;
476 4          return disem$offset;
477 4      end;
478 3      end;
479 2      end disem;

480 1      end disem86;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
124	0011H	2	A. WORD 125 126 127 128
45	0004H	2	A. WORD PARAMETER AUTOMATIC 46 47 48
26	0004H	2	A. WORD PARAMETER AUTOMATIC 27 28 29 30
			31
17			AHREG. LITERALLY
17			ALREG. LITERALLY 98
12	0000H	16	ALTTABLE BYTE BASED(ALTTABLEBASE) ARRAY(16) 420 421
12	000AH	2	ALTTABLEBASE WORD 12 419 420 421
12	0000H	16	ALTTABLEPTRS WORD ARRAY(8) EXTERNAL(0) 419
16			AXREG. LITERALLY 97 335
40	0004H	1	B. BYTE PARAMETER AUTOMATIC 41 42 43
34	0004H	1	B. BYTE PARAMETER AUTOMATIC 35 36 37 38
28	0000H	1	B. BYTE BASED(A) 29 30
17			BHREG. LITERALLY
17			BLREG. LITERALLY
14	01D5H	1	BORWFLAG BYTE 148 317 341 390 403 411 415
16			BPREG. LITERALLY
16			BXREG. LITERALLY
13	01CDH	1	BYTE1REGBITS BYTE 56 222 326 337 397
20	0000H	1	C. BYTE PARAMETER 21
17			CHREG. LITERALLY
17			CLREG. LITERALLY 321
23	0002H	11	COMMA. PROCEDURE STACK=0006H 272 277 286 291
			298 306 311 319 330 336 343 357 362 369
			380 392 398 405
20	0000H		CONOUT PROCEDURE EXTERNAL(2) STACK=0000H 24 30
			37 38 71 73 92 136 138 166 177 183
			188 202 205 252 260 322
2			CR LITERALLY
18			CSREG. LITERALLY
16			CXREG. LITERALLY
13	01CEH	1	DBIT BYTE 61 180 269 283 303 354
17			DHREG. LITERALLY
16			DIREG. LITERALLY
414	0694H	295	DIS. PROCEDURE STACK=0028H 475
459	07BBH	90	DISEM. PROCEDURE WORD PUBLIC STACK=0030H
1	0002H		DISEM36. PROCEDURE STACK=0000H
14	0000H	1	DISEMBYTE. BYTE BASED(DISEMPTR) ARRAY(1) 52 56 57
			58 59 60 61 105 118 119 120 125 157
			158 213 226 327 465 466
14	0012H	2	DISEMEND WORD
14	000EH	2	DISEMOFFSET. WORD A1 53 106 110 121 128 129 132
			133 139 142 159 164 207 216 258 476
14	000EH	4	DISEMPTR POINTER 14 52 56 57 58 59 60
			61 105 109 118 119 120 125 132 137 157
			158 163 213 226 257 327 462 465 466
14	0000H	2	DISEMWORD. WORD BASED(DISEMPTR) ARRAY(1) 109 132 137
			163 257
459	0004H	4	DISLOC POINTER PARAMETER AUTOMATIC 460 462

COPYRIGHT 1981, 1982

CROSS-REFERENCE

CROSS-REFERENCE

CROSS-REFERENCE

SER =

17		DLREG.	LITERALLY						
18		DSREG.	LITERALLY						
16		DXREG.	LITERALLY	305	312				
50	0077H	26 ERROR.	PROCEDURE STACK=0012H		457				
14	01D6H	1 ERRORFLAG. . . .	BYTE	218	350	375	386	415	424 426 456
18		ESREG.	LITERALLY						
2		FALSE.	LITERALLY	415					
		HIGH.	BUILTIN	47					
193	0200H	1 I.	BYTE	199	200				
15	0000H	512 INSTRTABLE. . . .	BYTE ARRAY(512) EXTERNAL(1)					465	466
14	01D4H	1 INSTRTYPE. . . .	BYTE	417	421	423	425	466	467 473 474
11	0000H	1 LEFTBRACKET. . . .	BYTE DATA	136	166				
193	01FFH	1 LEN.	BYTE	194	195	196	198	200	
2		LF.	LITERALLY						
		LOW.	BUILTIN	48	126				
14	01D3H	1 MNEUMONINDEX. . .	BYTE	195	198	419	420	465	
13	01CAH	1 MODBITS.	BYTE	57	143	150	155	161	373 384
8	01BFH	5 NOPS.	BYTE ARRAY(5) PUBLIC INITIAL						
461	0202H	1 NPREFIX.	BYTE	463	467	470			
9	01C4H	6 OPNIN.	BYTE ARRAY(6) PUBLIC INITIAL					195	198
3	0018H	24 OPS2.	BYTE ARRAY(24) INITIAL					10	
4	0030H	207 OPS3.	BYTE ARRAY(207) INITIAL					10	
5	00FFH	100 OPS4.	BYTE ARRAY(100) INITIAL					10	
6	0163H	80 OPS5.	BYTE ARRAY(80) INITIAL					10	
7	01B3H	12 OPS6.	BYTE ARRAY(12) INITIAL					10	
89	0159H	25 PRINT2REG16. . . .	PROCEDURE STACK=0018H		168	169	170	171	
75	0172H	26 PRINTAREG.	PROCEDURE STACK=0014H		273	276	287	290	
			297 307 310						
63	00DAH	24 PRINTBORW.	PROCEDURE STACK=000CH		149				
40	0045H	27 PRINTBYTE.	PROCEDURE STACK=000EH		47	48	52	105	
			158 397						
108	01B0H	20 PRINTDATA16. . . .	PROCEDURE STACK=0018H		114	248	259		
104	019DH	19 PRINTDATA8.	PROCEDURE STACK=0012H		115	244	285	292	
			370						
112	01C4H	20 PRINTDATA8OR16. . .	PROCEDURE STACK=001CH		299	331	344	393	
			408						
117	01D8H	45 PRINTDATASW. . . .	PROCEDURE STACK=0018H		407				
135	024BH	34 PRINTDIRECTADDR. . .	PROCEDURE STACK=0018H		152	271	278		
26	000DH	25 PRINTM.	PROCEDURE PUBLIC STACK=0008H			51	65	66	
192	0368H	116 PRINTMNEMONIC. . .	PROCEDURE STACK=0006H		210	215	221	227	
			231 235 239 243 247 251		256	264	268	282	
			296 302 316 328 334 340		353	367	378	389	
			396 402 469						
179	033AH	46 PRINTMODREGH. . . .	PROCEDURE STACK=0020H		265				
141	026DH	205 PRINTMODRM.	PROCEDURE STACK=001CH		184	187	240	318	
			342 358 361 368 381 391		399	404			
34	0026H	31 PRINTNIBBLE. . . .	PROCEDURE STACK=0008H		42	43			
68	00F2H	41 PRINTREG.	PROCEDURE STACK=000AH		77	81	102		
79	012CH	17 PRINTREG16.	PROCEDURE STACK=0010H		86	91	93	97	
			172 173 174 175 222 305		312	335	337	379	
75	011BH	17 PRINTREG8.	PROCEDURE STACK=0010H		87	98	321		
83	013DH	28 PRINTREG8OR16. . . .	PROCEDURE STACK=0016H		145	182	189	329	
100	018CH	17 PRINTSEGREG. . . .	PROCEDURE STACK=0010H		228	356	363		
131	0230H	27 PRINTSIGNED16. . . .	PROCEDURE STACK=0018H		236				
123	0205H	43 PRINTSIGNED8. . . .	PROCEDURE STACK=0018H		232				
45	0060H	23 PRINTWORD.	PROCEDURE PUBLIC STACK=0014H			109	119	120	
			128 132 137 163 261						

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

89	0003H	1	R1	BYTE PARAMETER AUTOMATIC	90	91
89	0004H	1	R2	BYTE PARAMETER AUTOMATIC	90	93
100	0004H	1	REG.	BYTE PARAMETER AUTOMATIC	101	102
79	0004H	1	REG.	BYTE PARAMETER AUTOMATIC	80	81
75	0004H	1	REG.	BYTE PARAMETER AUTOMATIC	76	77
68	0004H	1	REG.	BYTE PARAMETER AUTOMATIC	69	70
19	01D7H	16	REG16.	BYTE ARRAY(16) PUBLIC INITIAL		81
19	01E7H	16	REG8	BYTE ARRAY(16) PUBLIC INITIAL		77
68	0003H		REGADD	WORD PARAMETER AUTOMATIC	69	70
13	01CBH	1	REGBITS.	BYTE 58 182 189 326 329 348 356 363		
				379 397 420 421		
83	0004H	1	REGNUM	BYTE PARAMETER AUTOMATIC	84	86 87
11	0001H	1	RIGHTBRACKET	BYTE DATA 138 177		
13	01CCH	1	RMBITS	BYTE 59 145 150 157 167		
			ROL.	BUILTIN 118		
13	01CFH	1	SBIT	BYTE 61 406		
19	01F7H	8	SEGREG	BYTE ARRAY(8) PUBLIC INITIAL		102
55	0091H	73	SETBITS.	PROCEDURE STACK=0006H	325	416
			SHL.	BUILTIN 70 397		
			SHR.	BUILTIN 42 57 58 61 226 327		
16			SIREG.	LITERALLY		
16			SPREG.	LITERALLY		
18			SSREG.	LITERALLY		
14	0000H	1	TABLECHAR.	BYTE BASED(TABLEPTR) 71 73 202		
14	000CH	2	TABLEPTR	WORD 14 70 71 72 73 198 202 203		
10	0000H	10	TABPTRS.	WORD ARRAY(5) PUBLIC INITIAL		198
255	0013H	2	TEMP	WORD 257 261		
225	0201H	1	TEMP	BYTE 226 228		
2			TRUE	LITERALLY 218 317 341 347 350 375 386		
				390 403 411 424 426 464		
209	03DCH	8	TYPE1.	PROCEDURE STACK=000AH	427	
242	0451H	11	TYPE10	PROCEDURE STACK=0016H	434	
246	045CH	11	TYPE11	PROCEDURE STACK=001CH	435	
250	0467H	14	TYPE12	PROCEDURE STACK=000AH	436	
254	0475H	39	TYPE13	PROCEDURE STACK=001CH	437	
263	049CH	11	TYPE14	PROCEDURE STACK=0024H	438	
267	04A7H	35	TYPE18	PROCEDURE STACK=001CH	439	
212	03E4H	30	TYPE2.	PROCEDURE STACK=000AH	428	
281	04CAH	35	TYPE22	PROCEDURE STACK=0018H	440	
295	04EDH	17	TYPE23	PROCEDURE STACK=0020H	441	
301	04FEH	41	TYPE25	PROCEDURE STACK=0018H	442	
315	0527H	40	TYPE27	PROCEDURE STACK=0020H	443	
220	0402H	15	TYPE3.	PROCEDURE STACK=0014H	429	
324	054FH	46	TYPE31	PROCEDURE STACK=0020H	444	
333	057DH	24	TYPE33	PROCEDURE STACK=0014H	445	
339	0595H	22	TYPE34	PROCEDURE STACK=0020H	446	
346	05ABH	62	TYPE36	PROCEDURE STACK=0020H	447	
366	05E9H	17	TYPE38	PROCEDURE STACK=0020H	448	
372	05FAH	35	TYPE39	PROCEDURE STACK=0020H	449	
224	0411H	31	TYPE4.	PROCEDURE STACK=0014H	430	
383	061DH	36	TYPE40	PROCEDURE STACK=0020H	450	
395	0641H	29	TYPE42	PROCEDURE STACK=0020H	451	
401	065EH	41	TYPE44	PROCEDURE STACK=0020H	453	
410	0687H	13	TYPE45	PROCEDURE STACK=0024H	454	
230	0430H	11	TYPE5.	PROCEDURE STACK=001CH	431	
234	043BH	11	TYPE6.	PROCEDURE STACK=001CH	432	

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93955

SER. #

238	0446H	11	TYPE8.	PROCEDURE STACK=0020H	412	433
13	01D0H	1	VBIT	BYTE	61	320
13	01D1H	1	WBIT	BYTE	60	64 85 96 113 327 347 406
13	01D2H	1	ZBIT	BYTE	60	

MODULE INFORMATION:

CODE AREA SIZE = 0815H 2069D
CONSTANT AREA SIZE = 0016H 22D
VARIABLE AREA SIZE = 0203H 515D
MAXIMUM STACK SIZE = 0030H 48D
637 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE INSTR86
 OBJECT MODULE PLACED IN INSTR86.OBJ
 COMPILER INVOKED BY: PLM86 INSTR86.PL M DEBUG PAGERWIDTH(100) XREF

```
$title('instruction table for 8086 disassembler')
$date(10/5/80)
```

```
1 instr86: do;
```

```
2 1 declare
```

```
    qq$in literally '0',
    alt1 literally '0',
    alt2 literally '1',
    alt3 literally '2',
    alt4 literally '3',
    alt5 literally '4',
    alt6 literally '5',
    alt7 literally '6',
    alt8 literally '7';
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
$include(optab.lit)
```

```
3 1 = declare
```

```
= op2$in literally '0',
= op3$in literally '12',
= op4$in literally '81',
= op5$in literally '106',
= op6$in literally '122';
```

```
4 1 = declare
```

```
= AAA$in literally 'op3$in + 0',
= AAD$in literally 'op3$in + 1',
= AAM$in literally 'op3$in + 2',
= AAS$in literally 'op3$in + 3',
= ADC$in literally 'op3$in + 4',
= ADD$in literally 'op3$in + 5',
= AND$in literally 'op3$in + 6',
= CALL$in literally 'op4$in + 0',
= CALLF$in literally 'op5$in + 0',
= CBW$in literally 'op3$in + 7',
= CLC$in literally 'op3$in + 8',
= CLD$in literally 'op3$in + 9',
= CLI$in literally 'op3$in + 10',
= CMC$in literally 'op3$in + 11',
= CMP$in literally 'op3$in + 12',
= CMPSB$in literally 'op5$in + 1',
= CMPSW$in literally 'op5$in + 2',
= CS$in literally 'op3$in + 13',
= CWD$in literally 'op3$in + 14',
= DAA$in literally 'op3$in + 15',
= DAS$in literally 'op3$in + 16',
= DEC$in literally 'op3$in + 17',
= DIV$in literally 'op3$in + 18',
= DS$in literally 'op3$in + 19',
= ES$in literally 'op3$in + 20',
```

= ESC\$in literally 'op3\$in + 21',
= HLT\$in literally 'op3\$in + 22',
= IDIV\$in literally 'op4\$in + 1',
= INUL\$in literally 'op4\$in + 2',
= IN\$in literally 'op2\$in + 0',
= INC\$in literally 'op3\$in + 23',
= INT\$in literally 'op3\$in + 24',
= INTO\$in literally 'op4\$in + 3',
= IRET\$in literally 'op4\$in + 4',
= JA\$in literally 'op2\$in + 1',
= JAE\$in literally 'op3\$in + 25',
= JB\$in literally 'op2\$in + 2',
= JBE\$in literally 'op3\$in + 26',
= JC\$in literally 'op2\$in + 3',
= JCXZ\$in literally 'op4\$in + 5',
= JE\$in literally 'op2\$in + 4',
= JG\$in literally 'op2\$in + 5',
= JGE\$in literally 'op3\$in + 27',
= JL\$in literally 'op2\$in + 6',
= JLE\$in literally 'op3\$in + 28',
= JMP\$in literally 'op3\$in + 29',
= JMPF\$in literally 'op4\$in + 6',
= JMPS\$in literally 'op4\$in + 7',
= JNA\$in literally 'op3\$in + 30',
= JNAE\$in literally 'op4\$in + 8',
= JNB\$in literally 'op3\$in + 31',
= JNBE\$in literally 'op4\$in + 9',
= JNC\$in literally 'op3\$in + 32',
= JNE\$in literally 'op3\$in + 33',
= JNG\$in literally 'op3\$in + 34',
= JNGE\$in literally 'op4\$in + 10',
= JNL\$in literally 'op3\$in + 35',
= JNLE\$in literally 'op4\$in + 11',
= JNO\$in literally 'op3\$in + 36',
= JNP\$in literally 'op3\$in + 37',
= JNS\$in literally 'op3\$in + 38',
= JNZ\$in literally 'op3\$in + 39',
= JO\$in literally 'op2\$in + 7',
= JP\$in literally 'op2\$in + 8',
= JPE\$in literally 'op3\$in + 40',
= JPO\$in literally 'op3\$in + 41',
= JS\$in literally 'op2\$in + 9',
= JZ\$in literally 'op2\$in + 10',
= LAHF\$in literally 'op4\$in + 12',
= LDS\$in literally 'op3\$in + 42',
= LEA\$in literally 'op3\$in + 43',
= LES\$in literally 'op3\$in + 44',
= LOCK\$in literally 'op4\$in + 13',
= LODSB\$in literally 'op5\$in + 3',
= LODSW\$in literally 'op5\$in + 4',
= LOOP\$in literally 'op4\$in + 14',
= LOOPE\$in literally 'op5\$in + 5',
= LOOPNE\$in literally 'op6\$in + 0',
= LOOPNZ\$in literally 'op6\$in + 1',
= LOOPZ\$in literally 'op5\$in + 6',
= MOV\$in literally 'op3\$in + 45',
= MOVSB\$in literally 'op5\$in + 7',

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

= MOVSW$in literally 'op5$in + 8',
= MUL$in literally 'op3$in + 46',
= NEG$in literally 'op3$in + 47',
= NOP$in literally 'op3$in + 48',
= NOT$in literally 'op3$in + 49',
= OK$in literally 'op2$in + 11',
= OUT$in literally 'op3$in + 50',
= POP$in literally 'op3$in + 51',
= POPF$in literally 'op4$in + 15',
= PUSH$in literally 'op4$in + 16',
= PUSHF$in literally 'op5$in + 9',
= RCL$in literally 'op3$in + 52',
= RCR$in literally 'op3$in + 53',
= REP$in literally 'op3$in + 54',
= REPE$in literally 'op4$in + 17',
= RLPNE$in literally 'op5$in + 10',
= REPNZ$in literally 'op5$in + 11',
= REPZ$in literally 'op4$in + 18',
= RET$in literally 'op3$in + 55',
= RETF$in literally 'op4$in + 19',
= ROL$in literally 'op3$in + 56',
= ROR$in literally 'op3$in + 57',
= SAHF$in literally 'op4$in + 20',
= SAL$in literally 'op3$in + 58',
= SAR$in literally 'op3$in + 59',
= SBB$in literally 'op3$in + 60',
= SCASB$in literally 'op5$in + 12',
= SCASW$in literally 'op5$in + 13',
= SHL$in literally 'op3$in + 61',
= SHR$in literally 'op3$in + 62',
= SS$in literally 'op3$in + 63',
= STC$in literally 'op3$in + 64',
= STD$in literally 'op3$in + 65',
= STI$in literally 'op3$in + 66',
= STOSB$in literally 'op5$in + 14',
= STOSW$in literally 'op5$in + 15',
= SUB$in literally 'op3$in + 67',
= TEST$in literally 'op4$in + 21',
= WAIT$in literally 'op4$in + 22',
= XCHG$in literally 'op4$in + 23',
= XLAT$in literally 'op4$in + 24',
= XOR$in literally 'op3$in + 68';
= ;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

5 1 declare

```

type$0 literally '0',
type$1 literally '1',
type$2 literally '2',
type$3 literally '3',
type$4 literally '4',
type$5 literally '5',
type$6 literally '6',
type$7 literally '7',
type$8 literally '7',
type$9 literally '7',
type$10 literally '8',
type$11 literally '9',

```

```

type$12 literally '10',
type$13 literally '11',
type$14 literally '12',
type$15 literally '12',
type$16 literally '12',
type$17 literally '12',
type$18 literally '13',
type$19 literally '13',
type$20 literally '13',
type$21 literally '13',
type$22 literally '14',
type$23 literally '15',
type$24 literally '15',
type$25 literally '16',
type$26 literally '16',
type$27 literally '17',
type$28 literally '17',
type$29 literally '17',
type$30 literally '17',
type$31 literally '18',
type$32 literally '18',
type$33 literally '19',
type$34 literally '20',
type$35 literally '20',
type$36 literally '21',
type$37 literally '21',
type$38 literally '22',
type$39 literally '23',
type$40 literally '24',
type$41 literally '24',
type$42 literally '25',
type$43 literally '26',
type$44 literally '27',
type$45 literally '28';

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

6 1 declare
    prefix$type literally 'Offh';

7 1 declare alt$table$ptrs (8) address public data (
    .alt$1$tab,
    .alt$2$tab,
    .alt$3$tab,
    .alt$4$tab,
    .alt$5$tab,
    .alt$6$tab,
    .alt$7$tab,
    .alt$8$tab);

8 1 declare alt$1$tab (*) byte data (
    add$in, type$44,
    or$in, type$34,
    adc$in, type$44,
    sbb$in, type$44,
    and$in, type$34,
    sub$in, type$44,
    xor$in, type$34,
    cmp$in, type$44);

```

```
9 1 declare alt$2$tab (*) byte data (  
    add$in, type$44,  
    qq$in, type$0,  
    adc$in, type$44,  
    sbb$in, type$44,  
    qq$in, type$0,  
    sub$in, type$44,  
    qq$in, type$0,  
    cmp$in, type$44);
```

```
10 1 declare alt$3$tab (*) byte data (  
    add$in, type$38,  
    qq$in, type$0,  
    adc$in, type$38,  
    sbb$in, type$38,  
    qq$in, type$0,  
    sub$in, type$38,  
    qq$in, type$0,  
    cmp$in, type$38);
```

```
11 1 declare alt$4$tab (*) byte data (  
    rol$in, type$29,  
    ror$in, type$29,  
    rcl$in, type$29,  
    rcr$in, type$29,  
    shl$in, type$29,  
    shr$in, type$29,  
    qq$in, type$0,  
    sar$in, type$29);
```

```
12 1 declare alt$5$tab (*) byte data (  
    rol$in, type$27,  
    ror$in, type$27,  
    rcl$in, type$27,  
    rcr$in, type$27,  
    shl$in, type$27,  
    shr$in, type$27,  
    qq$in, type$0,  
    sar$in, type$27);
```

```
13 1 declare alt$6$tab (*) byte data (  
    test$in, type$34,  
    qq$in, type$0,  
    not$in, type$45,  
    neg$in, type$45,  
    mul$in, type$45,  
    imul$in, type$45,  
    div$in, type$45,  
    idiv$in, type$45);
```

```
14 1 declare alt$7$tab (*) byte data (  
    inc$in, type$45,  
    dec$in, type$45,  
    qq$in, type$0,  
    qq$in, type$0,  
    qq$in, type$0,
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

qq$in, type$0,
qq$in, type$0,
qq$in, type$0);

```

15 1 declare alt\$tab (*) byte data (

```

inc$in, type$45,
dec$in, type$45,
call$in, type$9,
callf$in, type$7,
jmp$in, type$9,
jmpf$in, type$7,
push$in, type$7,
qq$in, type$0);

```

/*

instruction table for 8086 disassembler
instruction is index into table
there are 2 bytes per instruction:

1. index into ascii opcode table
2. instruction type (how many operands of what type, etc.)

*/

16 1 declare instr\$table (512) byte public data (

```

add$in, type$14, /* 0 */
add$in, type$15,
add$in, type$16,
add$in, type$17,
add$in, type$23,
add$in, type$24,
push$in, type$4,
pop$in, type$4,
or$in, type$14,
or$in, type$15,
or$in, type$16,
or$in, type$17,
or$in, type$23,
or$in, type$24,
push$in, type$4,
qq$in, type$0,
adc$in, type$14, /* 10 */
adc$in, type$15,
adc$in, type$16,
adc$in, type$17,
adc$in, type$23,
adc$in, type$24,
push$in, type$4,
pop$in, type$4,
sbb$in, type$14,
sbb$in, type$15,
sbb$in, type$16,
sbb$in, type$17,
sbb$in, type$23,
sbb$in, type$24,
push$in, type$4,
pop$in, type$4,
and$in, type$14, /* 20 */
and$in, type$15,

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

and\$in, type\$16,
and\$in, type\$17,
and\$in, type\$23,
and\$in, type\$24,
es\$in, prefix\$type,
daa\$in, type\$1,
sub\$in, type\$14,
sub\$in, type\$15,
sub\$in, type\$16,
sub\$in, type\$17,
sub\$in, type\$23,
sub\$in, type\$24,
cs\$in, prefix\$type,
daa\$in, type\$1,
xor\$in, type\$14, /* 30 */
xor\$in, type\$15,
xor\$in, type\$16,
xor\$in, type\$17,
xor\$in, type\$23,
xor\$in, type\$24,
ss\$in, prefix\$type,
aaa\$in, type\$1,
cap\$in, type\$14,
cap\$in, type\$15,
cap\$in, type\$16,
cap\$in, type\$17,
cap\$in, type\$23,
cap\$in, type\$24,
ds\$in, prefix\$type,
aaa\$in, type\$1,
inc\$in, type\$3, /* 40 */
inc\$in, type\$3,
inc\$in, type\$3,
inc\$in, type\$3,
inc\$in, type\$3,
inc\$in, type\$3,
inc\$in, type\$3,
inc\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
dec\$in, type\$3,
push\$in, type\$3, /* 50 */
push\$in, type\$3,
push\$in, type\$3,
push\$in, type\$3,
push\$in, type\$3,
push\$in, type\$3,
push\$in, type\$3,
pop\$in, type\$3,
pop\$in, type\$3,
pop\$in, type\$3,

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
pop$in, type$3,
pop$in, type$3,
pop$in, type$3,
pop$in, type$3,
pop$in, type$3,
qq$in, type$0, /* 60 */
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
qq$in, type$0,
jo$in, type$5, /* 70 */
jno$in, type$5,
jb$in, type$5,
jnb$in, type$5,
jz$in, type$5,
jnz$in, type$5,
jbe$in, type$5,
ja$in, type$5,
js$in, type$5,
jns$in, type$5,
jpt$in, type$5,
jnp$in, type$5,
jl$in, type$5,
jnl$in, type$5,
jle$in, type$5,
jq$in, type$5,
alt1, type$43, /* 80 */
alt1, type$43,
alt2, type$43,
alt2, type$43,
test$in, type$14,
test$in, type$15,
xchg$in, type$16,
xchg$in, type$17,
mov$in, type$14,
mov$in, type$15,
mov$in, type$16,
mov$in, type$17,
mov$in, type$36,
lea$in, type$39,
mov$in, type$37,
pop$in, type$9,
nop$in, type$1, /* 90 */
xchg$in, type$33,
xchg$in, type$33,
xchg$in, type$33,
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950.

SER. #


```

int$in, type$10,
into$in, type$1,
iret$in, type$1,
alt4, type$43, /* D0 */
alt4, type$43,
alt5, type$43,
alt5, type$43,
aam$in, type$2,
aad$in, type$2,
qq$in, type$0,
xlat$in, type$1,
esc$in, type$42,
esc$in, type$42,
esc$in, type$42,
esc$in, type$42,
esc$in, type$42,
esc$in, type$42,
esc$in, type$42,
esc$in, type$42,
loopne$in, type$5, /* E0 */
loope$in, type$5,
loop$in, type$5,
jcxz$in, type$5,
in$in, type$22,
in$in, type$22,
out$in, type$22,
out$in, type$22,
call$in, type$6,
jmp$in, type$6,
japf$in, type$13,
jap$in, type$5,
in$in, type$25,
in$in, type$26,
out$in, type$25,
out$in, type$26,
lock$in, prefix$type, /* F0 */
qq$in, type$0,
repne$in, prefix$type,
rep$in, prefix$type,
hlt$in, type$1,
cmc$in, type$1,
alt6, type$43,
alt6, type$43,
clc$in, type$1,
stc$in, type$1,
cli$in, type$1,
sti$in, type$1,
cld$in, type$1,
std$in, type$1,
alt7, type$43,
alt8, type$43);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
4			AAAIN. LITERALLY 16
4			AADIN. LITERALLY 16
4			AAMIN. LITERALLY 16
4			AASIN. LITERALLY 16
4			ADCIN. LITERALLY 8 9 10 16
4			ADDIN. LITERALLY 8 9 10 16
2			ALT1 LITERALLY 16
8	0010H	16	ALT1TAB. BYTE ARRAY(16) DATA 7
2			ALT2 LITERALLY 16
9	0020H	16	ALT2TAB. BYTE ARRAY(16) DATA 7
2			ALT3 LITERALLY
10	0030H	16	ALT3TAB. BYTE ARRAY(16) DATA 7
2			ALT4 LITERALLY 16
11	0040H	16	ALT4TAB. BYTE ARRAY(16) DATA 7
2			ALT5 LITERALLY 16
12	0050H	16	ALT5TAB. BYTE ARRAY(16) DATA 7
2			ALT6 LITERALLY 16
13	0060H	16	ALT6TAB. BYTE ARRAY(16) DATA 7
2			ALT7 LITERALLY 16
14	0070H	16	ALT7TAB. BYTE ARRAY(16) DATA 7
2			ALT8 LITERALLY 16
15	0080H	16	ALT8TAB. BYTE ARRAY(16) DATA 7
7	0000H	16	ALTTABLEPTRS . . . WORD ARRAY(8) PUBLIC DATA
4			ANDIN. LITERALLY 8 16
4			CALLFIN. LITERALLY 15 16
4			CALLIN. LITERALLY 15 16
4			CSWIN. LITERALLY 16
4			CLCIN. LITERALLY 16
4			CLDIN. LITERALLY 16
4			CLIIN. LITERALLY 16
4			CNCIN. LITERALLY 16
4			CMPIN. LITERALLY 8 9 10 16
4			CMPSBIN. LITERALLY 16
4			CMPSWIN. LITERALLY 16
4			CSIN LITERALLY 16
4			CWDIN. LITERALLY 16
4			DAAIN. LITERALLY 16
4			DASIN. LITERALLY 16
4			DECIN. LITERALLY 14 15 16
4			DIVIN. LITERALLY 13
4			DSIN LITERALLY 16
4			ESWIN. LITERALLY 16
4			ESIN LITERALLY 16
4			HLTIN. LITERALLY 16
4			IDIVIN. LITERALLY 13
4			IMULIN. LITERALLY 13
4			INCIN. LITERALLY 14 15 16
4			ININ LITERALLY 16
1	0000H		INSTR84. PROCEDURE STACK=0000H

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

16	0090H	512	INSTRTABLE	BYTE ARRAY(512) PUBLIC DATA	
4			INTIN.	LITERALLY	16
4			INTOIN	LITERALLY	16
4			IRETIN	LITERALLY	16
4			JAEIN.	LITERALLY	
4			JAIN	LITERALLY	16
4			JBEIN.	LITERALLY	16
4			JBIN	LITERALLY	16
4			JCIN	LITERALLY	
4			JCXZIN	LITERALLY	16
4			JEIN	LITERALLY	
4			JGEIN.	LITERALLY	
4			JGIN	LITERALLY	16
4			JLEIN.	LITERALLY	16
4			JLIN	LITERALLY	16
4			JMPFIN	LITERALLY	15 16
4			JMPIN.	LITERALLY	15 16
4			JMP SIN	LITERALLY	16
4			JNAEIN	LITERALLY	
4			JNAIN.	LITERALLY	
4			JNBEIN	LITERALLY	
4			JNBIN.	LITERALLY	16
4			JNCIN.	LITERALLY	
4			JNEIN.	LITERALLY	
4			JNGEIN	LITERALLY	
4			JNGIN.	LITERALLY	
4			JNLEIN	LITERALLY	
4			JNLIN.	LITERALLY	16
4			JNOIN.	LITERALLY	16
4			JNPIN.	LITERALLY	16
4			JNSIN.	LITERALLY	16
4			JNZIN.	LITERALLY	16
4			JOIN	LITERALLY	16
4			JPEIN.	LITERALLY	
4			JPIN	LITERALLY	16
4			JPOIN.	LITERALLY	
4			JSIN	LITERALLY	16
4			JZIN	LITERALLY	16
4			LAHF IN	LITERALLY	16
4			LDSIN.	LITERALLY	16
4			LEAIN.	LITERALLY	16
4			LESIN.	LITERALLY	16
4			LOCKIN	LITERALLY	16
4			LODSBIN.	LITERALLY	16
4			LODSWIN.	LITERALLY	16
4			LOOPEIN.	LITERALLY	16
4			LOOPIN	LITERALLY	16
4			LOOPNEIN	LITERALLY	16
4			LOOPNZIN	LITERALLY	
4			LOOPZIN.	LITERALLY	
4			MOVIN.	LITERALLY	16
4			MOVSBIN.	LITERALLY	16
4			MOVSWIN.	LITERALLY	16
4			MULIN.	LITERALLY	13
4			NEGIN.	LITERALLY	13
4			NOPIN.	LITERALLY	16
4			NOTIN.	LITERALLY	13

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

3	OP2IN.	LITERALLY	8	16										
3	OP3IN.	LITERALLY	8	9	10	11	12	13	14					
		15 16												
3	OP4IN.	LITERALLY	13	15	16									
3	OP5IN.	LITERALLY	15	16										
3	OP6IN.	LITERALLY	16											
4	ORIN.	LITERALLY	8	16										
4	OUTIN.	LITERALLY	16											
4	POPFIN.	LITERALLY	16											
4	POPIN.	LITERALLY	16											
6	PREFIXTYPE	LITERALLY	16											
4	PUSHFIN.	LITERALLY	16											
4	PUSHIN.	LITERALLY	15	16										
2	QQIN.	LITERALLY	9	10	11	12	13	14	15					
		16												
4	RCLIN.	LITERALLY	11	12										
4	RCRIN.	LITERALLY	11	12										
4	REPEIN.	LITERALLY												
4	REPIN.	LITERALLY	16											
4	REPNEIN.	LITERALLY	16											
4	REPNZIN.	LITERALLY												
4	REPZIN.	LITERALLY												
4	RETFIN.	LITERALLY	16											
4	RETIN.	LITERALLY	16											
4	ROLIN.	LITERALLY	11	12										
4	RORIN.	LITERALLY	11	12										
4	SAHFIN.	LITERALLY	16											
4	SALIN.	LITERALLY												
4	SARIN.	LITERALLY	11	12										
4	SBBIN.	LITERALLY	8	9	10	16								
4	SCASBIN.	LITERALLY	16											
4	SCASWIN.	LITERALLY	16											
4	SHLIN.	LITERALLY	11	12										
4	SHRIN.	LITERALLY	11	12										
4	SSIN.	LITERALLY	16											
4	STCIN.	LITERALLY	16											
4	STDIN.	LITERALLY	16											
4	STIIN.	LITERALLY	16											
4	STOSBIN.	LITERALLY	16											
4	STOSWIN.	LITERALLY	16											
4	SUBIN.	LITERALLY	8	9	10	16								
4	TESTIN.	LITERALLY	13	16										
5	TYPE0.	LITERALLY	9	10	11	12	13	14	15					
		16												
5	TYPE1.	LITERALLY	16											
5	TYPE10.	LITERALLY	16											
5	TYPE11.	LITERALLY	16											
5	TYPE12.	LITERALLY	16											
5	TYPE13.	LITERALLY	16											
5	TYPE14.	LITERALLY	16											
5	TYPE15.	LITERALLY	16											
5	TYPE16.	LITERALLY	16											
5	TYPE17.	LITERALLY	16											
5	TYPE18.	LITERALLY	16											
5	TYPE19.	LITERALLY	16											
5	TYPE2.	LITERALLY	16											
5	TYPE20.	LITERALLY	16											

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

5	TYPE21	LITERALLY	16	
5	TYPE22	LITERALLY	16	
5	TYPE23	LITERALLY	16	
5	TYPE24	LITERALLY	16	
5	TYPE25	LITERALLY	16	
5	TYPE26	LITERALLY	16	
5	TYPE27	LITERALLY	12	
5	TYPE28	LITERALLY		
5	TYPE29	LITERALLY	11	
5	TYPE3.	LITERALLY	16	
5	TYPE30	LITERALLY		
5	TYPE31	LITERALLY	16	
5	TYPE32	LITERALLY	16	
5	TYPE33	LITERALLY	16	
5	TYPE34	LITERALLY	8	13
5	TYPE35	LITERALLY		
5	TYPE36	LITERALLY	16	
5	TYPE37	LITERALLY	16	
5	TYPE38	LITERALLY	10	
5	TYPE39	LITERALLY	16	
5	TYPE4.	LITERALLY	16	
5	TYPE40	LITERALLY	16	
5	TYPE41	LITERALLY	16	
5	TYPE42	LITERALLY	16	
5	TYPE43	LITERALLY	16	
5	TYPE44	LITERALLY	8	9
5	TYPE45	LITERALLY	13	14 15
5	TYPE5.	LITERALLY	16	
5	TYPE6.	LITERALLY	16	
5	TYPE7.	LITERALLY	15	
5	TYPE8.	LITERALLY		
5	TYPE9.	LITERALLY	15	16
4	WAITIN	LITERALLY	16	
4	XCHGIN	LITERALLY	16	
4	XLATIN	LITERALLY	16	
4	XORIN.	LITERALLY	8	16

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

MODULE INFORMATION:

CODE AREA SIZE = 0000H 0D
 CONSTANT AREA SIZE = 0290H 656D
 VARIABLE AREA SIZE = 0000H 0D
 MAXIMUM STACK SIZE = 0000H 0D
 560 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION